Suitability of a Neural Network to Identify Musical Patterns

To what extent is a One-Dimensional Convolutional Neural Network suitable to classify raags in Hindustani classical music?

Computer Science

Machine Learning

Word Count: 3541

Abstract

This investigation explores how suitable a one-dimensional convolutional neural network is for the task of classifying multiple Hindustani classical music *raags*. A one-dimensional convolutional neural network relies on various algorithms to recognize prominent features in a set of data, key to classifying datasets. The underlying methodology behind a neural network is particularly applicable to the classification of raags in Indian classic music, as each *raag* is distinguished by certain patterns and distinct movements of notes. This identification of patterns is what an aforementioned neural network structure is primarily built for. Based upon this elementary approach, an experiment was carried out involving the creation of a one-dimensional convolutional neural network to test the accuracy of a multi-class classifier between five different raags. The machine learning model was evaluated utilizing 10-fold cross validation with a complete confusion matrix combined across all folds. The results of this experiment found that though a one-dimensional neural network proved to be an accurate classification method between the five chosen raags, a two-dimensional network or alternate approach would be needed to classify a greater number of raags or classify raags from a more complex dataset.

Contents

1 Introduction

Recent research has explored the idea of identifying patterns within audio data. Specifically, machine learning and neural networks have been applied to classify musical genres. Based upon this well-developed usage of deep learning, the research question ‘To what extent is a One-Dimensional Convolutional Neural Network suitable to classify raags in Hindustani classical music?’ has been chosen in the interest of evaluating the feasibility of a neural network to distinguish between complex patterns in Indian music.

1.1 Music Classification

Researchers have investigated the utility of machine learning to classify music for years. Many projects have previously attempted to classify musical genres using deep learning specifically, such as identifying popular music versus rhythm and blues music [1]. These projects have been largely successful, leading to the question of classifying music from other cultures, such as Hindustani classical music. While Western music has these different genres that could potentially be classified computationally, at a first glance, one of the main aspects of Indian classical music that could be classified in this way is the different *raags*. Raags are often compared to their Western equivalent of scales, a concept fundamental to all music since the times of ancient Greece [2]. In the same way that Western music includes major, minor, and pentatonic scales, there are ten parent scales in Indian classical music. These parent scales, known as *thaats*, are what *raags* are derived from. From nearly two thousand years ago, when Indian classical music was developed in Vedic hymns of ancient Hindu temples, musicians have been creating many different raags based upon the ten original thaats [2]. Though the parallel between raags and scales is quite accurate, raags differ slightly from scales in that the ascending and descending scales of a raag differ from each other, while a Western scale is the same whether ascending or descending. Unlike Western music, two raags can often have the same sets of notes with extremely unique ascending and descending scales. Moreover, while Western scales are generally played with uniform emphasis on each notes, different raags are differentiated by their individual notes of focus and distinctive sequences or ‘moods’ [3]. If a particular sequence from one raag is used in another raag, it can sound completely out of place, especially to a trained musician. These patterns stand out as a unique aspect of these raags in Hindustani classical music, attracting the attention of researchers to tackle the issue of classifying these raags computationally. Certain challenges lie in classifying this type of music, as it requires consideration of both intrinsic properties, such as notes, and extrinsic properties, such as emotions conveyed [4]. Yet, because the idea of identifying patterns within data has been easily implemented in other genres of music using a convolutional neural network, applying a similar technique in this field must yield a certain level of success [5].

1.2 Deep Learning

First and foremost, the concept of deep learning must be explored in overarching terms to gain a greater understanding of its applicability to classification problems within music. The concept of deep learning implies modelling data with a computational system combining non-linear transformations [5]. Like their parent field of machine learning, deep learning models rely on inputs of training data and test data. Neural networks utilize the training data by processing these inputs in a structure relying on hidden layers of neurons. All inputs are ‘weighted’ individually and passed into an activation function, which varies depending on set parameters chosen by the user, resulting in an output [5]. Essentially, within the first few layers of a deep learning network, the neurons identify simple patterns within the input. Within higher layers, these base patterns are then used to form more complex patterns within until the network arrives at a final classification. The final classification of the network implemented in this investigation outputs a set of respective probabilities that a certain input variable might correlate to each potential output. Deep learning includes deep neural networks, recurrent neural networks, and convolutional neural networks. It has been applied to many fields include natural language processing, speech recognition, as well as audio recognition, which will be the primary pursuit of this investigation [5].

1.3 One-Dimensional Convolutional Neural Networks

Knowing that chosen raags will have certain patterns and emphasized notes that distinguish them from one another, it is now imperative to identify how a one-dimensional convolutional neural network could specifically be applied to the fascinating challenge of the classification of Indian raags. Previous research has explored a recurrent neural network, hybrid architecture with a paralleled convolutional neural network and recurrent neural network, as well as Bayesian Net and Naive Bayes classifiers [3]. However, for the scope of this project, a one-dimensional convolutional neural network seemed most efficient to create, train, and evaluate. The methodology of a one-dimensional network for this project rather than a two or three-dimensional neural network must be appreciated. In most research, two-dimensional convolutional neural networks are often implemented for image recognition [6]. Recurrent neural networks have previously been implemented in classifying audio data between Experimental, Folk, Hip-Hop and Instrumental genres [1]. In this project, audio files were converted into spectrograms, or representations of the frequencies of the file. Two-dimensional convolutional neural networks could be applied for this classification, as they are suited for the RGB inputs of image processing projects. However, one-dimensional networks are different, as rather than classifying images, they are more often used for tasks such as natural language processing [6].

These networks would not be as applicable in image recognition, though they are often used in building a classifier that relies on numerical data inputs. Moreover, one-dimensional networks are especially effective in finding features from short segments of a dataset [6]. As stated before, raags are often distinguished by certain recurring sequences that are individual and distinct for each raag. Thus, a one-dimensional convolutional neural network proves to especially useful in classifying raags, as the patterns of the melody within shorter, fixed-length portions of the dataset are more significant for classification rather than the location of a particular point in the audio dataset does not matter as it might in a neural network with a different number of dimensions [7].

1.4 Discussion of Investigative Scope

The earlier introduction to neural networks describes their applicability to the classification of images. Yet, in this investigation, the constructed neural network must be applied to classifying audio data. Audio data can be manifested in many ways, including images such as spectrograms, which describes frequency versus time on a color-based plot, but the use of a one-dimensional convolutional network confines the input to numerical data [1]. However, many options lie within numerical data as well. The frequency, pitch, or amplitude of the sound could be used to identify patterns in raags. By a simple process of elimination, the amplitude of the audio is not as relevant to raag classification, as raags must be classified based upon melodic patterns rather than changes in the volume of the audio. Though volume often reflects emotional changes, identifying emotional variation computationally is far too extensive for the scope of this project. Eliminating amplitude data leaves pitch and frequency. Translating audio data into frequency versus time requires a Fast Fourier Tranformation (FFT), which could potentially decrease the efficiency of a machine learning model with large audio files [9]. Thus, the one option left is to translate audio files into a numerical array of pitches before a one-dimensional convolutional neural network could be applied to it. Having identifying the type of data that will be used, further difficulties and limitations lie within the necessity for a simple dataset in the case of a one-dimensional convolutional neural network. Because pitches need to be identified from the audio file, any computational library would not function properly if the audio file in question has an excessive amount of ornamentation or a lack of clarity. This issue is especially prominent in the field of Hindustani classical music, as the majority of raags in Indian music are often sung with accompanying instruments, such as the tabla or a harmonium. These conflicting layers of instrumental versus vocal music within potential datasets could make it extremely difficult to obtain numerical data. Thus, the input data must be limited to data that is solely vocal, without instrumental supplements that could interfere with the main, necessary pitches, or data that is solely instrumental. Lastly, the scope of the investigation was defined before creating the neural network. With an extremely expansive set of variations over its history, Hindustani classical music contains many raags that have extremely similar patterns and are even difficult for a trained musician's ear to distinguish between, much less a computational structure, no matter how complex it is [8]. Thus, the investigation question was limited to a certain number of raags – in this case, five raags were chosen. To cover the broadest scope within the comprehensive tree of Hindustani raags, the five raags that were chosen derived from five of the aforementioned parent *thaats*. These raags will be described in further detail in the description of the creation of the dataset.

2 Investigation

Having realized the scope, necessities, and potential limitations of classifying Hindustani raags with deep learning, an experiment was conducted to investigate the suitability of a one-dimensional neural network to classify five Hindustani raags. This experiment firstly required the creation of an unbiased, reasonably-sized dataset. Next, a one-dimensional convolutional neural network was created with Python to test the efficiency and accuracy of the said network in identifying the various raags that the dataset was aimed towards. Lastly, the model was evaluated and analyzed utilizing cross-fold validation.

2.1 Dataset Creation

As stated before, the scope of this investigation was limited to five raags. To take full advantage of the potential of the one-dimensional convolutional neural network, the five raags that were chosen had extremely differing patterns that would be extremely easy to distinguish [2]. Moreover, the raags were derived from the primary parent scales, ensuring that they covered a large number of compositions within Indian classical music. The names of the five raags chosen are listed below in Figure 1 with their corresponding ascending and descending scales in Western music notation:

Raag Name	**Western music notation (ascending)**	**Western music notation (descending)**
Aasavari	C D F G G# C	C A# G# G F D# D C
Bhairav	C D# E F G G# B C	C B G# G F E D# C
Bilawal	C D E F G A B C	C B A G F E D C
Khamaaj	C E F G A B C	C A# A G F E D C
Yaman	B D E F# A B C	C B A G F# E D C

Figure 1: Names of five raags with their scales.

Next, a sufficiently large dataset was created for each of these raags. The dataset was derived from audio files without instrumental music in the background, as instrumental music would interfere with a library attempting to extract pitches from the audio file. Principally, the audio files was a long string of pitches with minimal ornamentation to make the network as accurate as possible. Moreover, the dataset was further limited to one voice and was thus self-created. The usage of different voices in the dataset could cause excessive complexity within the one-dimensional network, as varying timbres and keys of voices would change the numerical array of pitches and cause the network to potentially misidentify certain files. Thus, short audio passages of raag variations were recorded to create the dataset. Each audio file was approximately 5-7 seconds long, with a few being 'catchphrases' or significant sequences that often manifested themselves within the raag in question. Twenty samples were recorded for each raag, with the same voice for all samples. These specifications of the dataset ensured that the neural network would be able to easily distinguish the audio files based on solely the varying pitches rather than needing to analyze a large number of complex factors.

2.2 Dataset Processing

Following the creation of this audio set of one hundred files in the Waveform Audio File Format, these files were transferred into numerical data easily readable by the future constructed neural network. For this task, the Python library 'aubio,' was used [10]. This library included a number of efficient tools for audio analysis, including pitch tracking through fundamental frequency estimation. A program was built using this library (see appendix) that iterated through each WAV file utilizing the Pandas Python library. After reading the audio file, the program determined the file's sample rate. Based upon this sample rate, aubio was used to compile an array of pitches for each sample and then save this array as a comma-separated values files (CSV file). Furthermore, a pitch of 'confidences' was generated for each file, indicating the uncertainty for each pitch that the library determined from the audio. Transforming the audio files to this type of numerical data concluded the creation of the necessary dataset.

2.3 Model Creation

A base one-dimensional convolutional neural network was then created in Python utilizing the Keras and Scikit.learn libraries in Python, two common deep learning libraries. The network was coded with multiple one-dimensional convolutional layers. Each of these layers identified key features from the original data and applied RELU activation. Between layers, one-dimensional max pooling was performed, preventing potential over-fitting of the model. Lastly, the final output layer was a dense layer with Sigmoid activation. Dropout was used before the final output layer to again reduce over-fitting. The batch size for the model was set to the default, providing faster training with mini-batches. The number of epochs were defined by increasing until the validation accuracy began decreasing at the same time that training accuracy increased, providing an indication of overfitting. The learning rate hyperparameter was chosen by beginning with a low learning rate and increasing it for every batch. The learning rate and training loss were graphed to find the point with the fastest loss decrease and this point defined the learning rate. The network was coded to output a 5x1 vector, where each value corresponded to the confidence score that the array belonged to one of the five classes, or raags, which are listed in Table 1.

2.4 Evaluation of Model

Following the implementation of this deep learning model, results were evaluated using 10-fold cross validation, a method in machine learning to "estimate the skill of a machine learning model on unseen data." [11]. No single fold contained segments of audio data produced from the same audio file, ensuring that cross-validation scores provided a fair estimate of the model's performance, described by the Table 2. A confusion matrix was additionally as a performance measurement for the model and can be viewed in Table 3.

Fold	**Accuracy**
1	0.94
2	0.97
3	0.97
4	0.98
5	0.94
6	0.96
7	0.95
8	0.93
9	0.87
10	0.94
Average	**0.95**

Table 2: Cross-fold validation

	Aasavari	Bhairav	Bilawal	Khamaaj	Yaman	Total
Aasavari	**17**	2	0	1	0	20
Bhairav	3	**16**	1	0	0	20
Bilawal	0	1	**17**	2	0	20
Khamaaj	0	1	3	**16**	0	20
Yaman	0	0	0	0	**20**	20

Table 3: Results of audio classification model

3 Discussion

Based upon this investigation, certain challenges were found that could be addressed in a more extensive research project and certain useful conclusions were found as well.

3.1 Limitations

The main limitations of this experiment were within the creation of the dataset. Yet, the dataset itself was preemptively limited to avoid the possibility of an extremely inaccurate neural network. Many challenges were found within the dataset, namely the necessity for:

1. A solely vocal or solely instrumental dataset,
2. Short audio files to ensure an efficient deep learning model,
3. All audio files being created by the same voice to avoid misclassifications of files with voices in different timbres or different keys.

However, the ideal dataset would not be created individually; rather, it would compile Indian music from various sources, including instrumental files, vocal files, and a combination of both files. This limitation could be alleviated by an approach besides a one-dimensional network, which will be elaborated upon later.

Another limitation of the experiment was the one-dimensional network itself. Creating spectrograms of the audio data for a two-dimensional network could have increased the model's potential for identifying music from varying sources and would increase its accuracy with a larger dataset.

3.2 Conclusions

The constructed one-dimensional convolutional neural network yielded a high accuracy and thus was highly suitable for this particular dataset. In comparison to previous research that has yielded accuracy within the 80% range, the 95% accuracy calculated with the confusion matrix suggests that this experiment was largely successful.

The chosen hyperparameters were clearly effective, as the model ran efficiently and moreover the accuracy clearly demonstrated that overfitting did not occur, suggesting that the number of epochs had been chosen properly.

As shown in Table 3 above, the raag classified most accurately was raag Yaman, as this raag was the only one that included the note F#, distinguishing it from the other raags. Bilawal and Khamaaj seemed to be misclassified for each other, as both raags included the similar C, E, F, G, A, B and C notes in their ascending scale. Aasavari and Bhairav displayed a similar trend, as they both contained the note of G# that the other raags did not contain. This suggests that the model depends upon these distinguishing notes to classify between raags. For a greater number of raags, it may have had less accuracy, as more raags would have these distinguishing notes, causing misclassification.

3.3 Extensions

As stated above, the dataset itself was very limited and highly biased; thus, this model would need to be further developed for a more variant dataset. Potential extensions of this experiment could involve used Fast Fourier Transform on audio data and creating a neural network based upon the change in frequency over time [9]. This change in frequency would be a potentially accurate and efficient method of detecting patterns within the music, relevant to the classification of these raags. This methodology would be compatible with different voices in different keys, as analyzing the change in frequency would not require a uniform voice as this neural network did. Moreover, adding raags that have a similar set of notes, but different patterns, could be another interesting problem to further explore, as this could drastically change the accuracy of the neural network. The accuracy of this experiment proves that a neural network can be applied to differentiate between musical patterns. This creates implications regarding the potential of computer science to decompose art creations as a whole. In the end, raags are a complex, unpredictable product of human thought and creativity and the accuracy of this experiment in distinguishing these human ideological creations proves the power of deep learning and neural networks in specific.

References

[1] P. Dwivedi, “Using CNNs and RNNs for Music Genre Recognition,” *Medium*, 27-Mar-2019. [Online]. Available: https://towardsdatascience.com/using-cnns-and-rnns-for musicgenre-recognition-2435fb2ed6af.

[2] Sadhana, “What is a Raga?,” *Raag Hindustani*. [Online]. Available: https://raag hindustani.com/Scales1.html.

[3] E. Patel and S. Chauhan, “Raag detection in music using supervised machine learning approach,” *International Journal of Advanced Technology and Engineering Exploration*, vol. 4, no. 29, pp. 58–67, Aug. 2017.

[4] Ross, Joe Cheri, et al. “Identifying Raga Similarity Through Embeddings Learned From Compositions' Notation.” *IBM Research India*, pp. 515–522.

[5] “Neural Networks and Introduction to Deep Learning.” *WikiStat.*

[6] N. Ackermann, “Introduction to 1D Convolutional Neural Networks in Keras for Time Sequences,” *Medium*, 14-Sep-2018.

[7] V. Kumar, H. Pandya, and C. Jawahar, “Identifying Ragas in Indian Music,” 2014 22nd *International Conference on Pattern Recognition*, 2014.

[8] “Indian classical music roots and ragas - Times of India,” *The Times of India*, 29-Sep-2016. [Online]. Available: https://timesofindia.indiatimes.com/Indian-classical-music-roots and-ragas/articleshow/54540906.cms.

[9] Sharma, Hiteshwari, and Rasmeet S. Bali. “Comparison of ML Classifiers for Raga Recognition.” vol. 5, *International Journal of Scientific and Research Publications*, 2015, pp. 1–5.

[10] “aubio,” PyPI. [Online]. Available: https://pypi.org/project/aubio/.

[11] J. Brownlee, “A Gentle Introduction to k-fold Cross-Validation,” Machine Learning Mastery, 08-Aug-2019. [Online]. Available: https://machinelearningmastery.com/k-fold cross-validation/.

[12] S. Narkhede, “Understanding Confusion Matrix,” Medium, 29-Aug-2019. [Online]. Available: https://towardsdatascience.com/understanding-confusion-matrix a9ad42dcfd62.

Appendix

Code for Data Processing:

```
import sys
import numpy as np
import pandas as pd
import os
from aubio import source, pitch

win_s = 2048
hop_s = 20

raags = ["aasavari", "bhairav", "bilawal", "khamaaj", "yaman"]

for filename in os.listdir("C:/Users/Administrator/Desktop/ee model/aasavari/"):
    if filename.endswith(".wav"):
        data = pd.read_csv("C:/Users/Administrator/Desktop/ee model/aasavari/" +
filename)
        newName = filename.split('.wav')

        s = source(filename, 44100, hop_s)
        samplerate = s.samplerate
        tolerance = 0.8

        pitch_o = pitch("yin", win_s, hop_s, samplerate)
        pitch_o.set_unit("midi")
        pitch_o.set_tolerance(tolerance)
        pitches = []
        confidences = []
        total_frames = 0
        while True:
            samples, read = s()
            pitch = pitch_o(samples)[0]
            pitches += [pitch]
            confidence = pitch_o.get_confidence()
            confidences += [confidence]
            total_frames += read
            if read < hop_s: break

        pd.DataFrame(np.array(pitches)).to_csv("C:/Users/Administrator/Desktop/ee
 model/aasavari/" + newName[0] + ".csv")
        print(np.array(pitches))
        print(len(pitches))
        print()
```

One-Dimensional Convolutional Neural Network Code:

```
import faulthandler
faulthandler.enable()

import glob
import os
import numpy as np
import pandas as pd
import json
import progressbar

from scipy.fftpack import fft, fftfreq
from scipy.io import wavfile

from sklearn.model_selection import train_test_split
from sklearn.metrics import precision_recall_curve, precision_score, recall_score
, accuracy_score, roc_auc_score, roc_curve

from AudioVisualization import plot_time_series, plot_fft, return_fft

from keras.models import Sequential, load_model
from keras.layers import Reshape, Conv1D, MaxPooling1D, GlobalAveragePooling1D, D
ropout, Dense, Conv2D, MaxPool2D, Flatten
from keras.callbacks import EarlyStopping
from keras.optimizers import Adam
from keras.preprocessing.image import ImageDataGenerator
from PIL import Image
from keras.layers.pooling import MaxPool2D


# hyperparameters
LEARNING_RATE = 10 ** (np.random.rand() - 3.5)
LEARNING_RATE = 0.0007
BATCH_SZ = 32
EPOCHS = 15

print("LEARNING RATE: ", LEARNING_RATE)
print("BATCH SIZE: ", BATCH_SZ)
print("EPOCHS: ", EPOCHS)

# one-hot encoding for multi-class classification
def encode_multi(y):
    y = y.lower()
    if y[len(y)-1] ==" ": y = y[:-1]
```

```python
    if y == "aasavari": return [1, 0, 0, 0, 0]
    elif y == "bhairav": return [0, 1, 0, 0, 0]
    elif y == "bilawal": return [0, 0, 1, 0, 0]
    elif y == "khamaaj": return [0, 0, 0, 1, 0]
    elif y == "yaman": return [0, 0, 0, 0, 1]
    else: print("encode_multi ERROR: ", y)

x = []
y = []
x = {'aasavari' : [], 'bhairav' : [], 'bilawal' : [], 'khamaaj' : [], "yaman" : [
]}
index_map = ["aasavari", "bhairav", "bilawal", "khamaaj", "yaman"]

for k in range(0, len(index_map)):
    cur = "C:/Users/Administrator/Desktop/ee model/"
    cur = cur + index_map[k] + "/*.wav"
    for f in glob.glob(cur):
        x[index_map[k]].append(f)

idx = 0

for raag in index_map:
    print(raag)
    for f in x[raag]:
        sample_rate, signal = wavfile.read(f)

        max_start_idx = 0
        max_start = 0
        max_end_idx = 0
        max_end = 0

        for i in range(len(signal)//2):
            if (signal[i] > max_start).any():
                max_start = signal[i]
                max_start_idx = i

        for i in range(len(signal)//2, len(signal)):
            if (signal[i] > max_end).any():
                max_end = signal[i]
                max_end_idx = i

        offset_start = 1
        offset_end = 2
        max_start_idx += offset_start * sample_rate
        max_end_idx -= offset_end * sample_rate
        segment_len = 3

        start = max_start_idx

        while start + segment_len * sample_rate < max_end_idx:

            fft_vals, fft_freqs = return_fft(sample_rate, signal[start: start+segment_len*sample_rate])

            fft_domain = [idx for idx, val in enumerate(fft_freqs) if val>9900 and val<10100]

            if idx==0:
                print(fft_domain)
                print(fft_vals[fft_domain])

            x[raag][f] = (signal[start: start+segment_len*sample_rate:40])

x_train = []
x_test = []
x_val = []
y_train = []
y_test = []
y_val = []

for i in range(5):
    #one-hot encoding
    y_encoded_multi = encode_multi(index_map[i])
    slice = int(0.7*len(x[index_map[i]]))
    slice2 = int(0.8*len(x[index_map[i]]))
    x_train.extend(x[index_map[i]][:slice])
    y_train.extend([y_encoded_multi] * slice)

    x_val.extend(x[index_map[i]][slice: slice2])
    y_val.extend([y_encoded_multi] * (slice2-slice+1))

    x_test.extend(x[index_map[i]][slice2:len(x[index_map[i]])])
    y_test.extend([y_encoded_multi] * (len(x[index_map[i]])-slice2+1))

x_train = np.array(x_train)
x_val = np.array(x_val)
x_test = np.array(x_test)

y_train = np.array(y_train)
```

```
y_val = np.array(y_val)
y_test = np.array(y_test)

print("x train shape ", x_train.shape)
p = np.random.permutation(len(x_train))
x_train = x_train[p]
y_train = y_train[p]

p = np.random.permutation(len(x_val))
x_val = x_val[p]
y_val = y_val[p]

p = np.random.permutation(len(x_test))
x_test = x_test[p]
y_test = y_test[p]

#Construct Model
model = Sequential()
model.add(Conv1D(64, 3, activation='relu', input_shape = (48000*3/40, 1)))
model.add(Conv1D(64, 3, activation='relu'))
model.add(MaxPooling1D(3))

model.add(Conv1D(128, 3, activation='relu'))
model.add(Conv1D(128, 3, activation='relu'))
model.add(MaxPooling1D(3))

model.add(Conv1D(256, 3, activation='relu'))
model.add(Conv1D(256, 3, activation='relu'))
model.add(Conv1D(256, 3, activation='relu'))
model.add(MaxPooling1D(3))

model.add(Conv1D(512, 3, activation='relu'))
model.add(Conv1D(512, 3, activation='relu'))
model.add(Conv1D(512, 3, activation='relu'))
model.add(MaxPooling1D(3))

model.add(Conv1D(1024, 3, activation='relu'))
model.add(Conv1D(1024, 3, activation='relu'))
model.add(Conv1D(1024, 3, activation='relu'))
model.add(Conv1D(1024, 3, activation='relu'))
model.add(MaxPooling1D(3))

model.add(Conv1D(1024, 3, activation='relu'))
model.add(Conv1D(1024, 3, activation='relu'))
model.add(Conv1D(1024, 3, activation='relu'))
```

```python
model.add(Conv1D(1024, 3, activation='relu'))
model.add(MaxPooling1D(3))

model.add(GlobalAveragePooling1D())

model.add(Dense(4096, activation='relu'))
model.add(Dropout(rate=0.5))
model.add(Dense(4096, activation='relu'))
model.add(Dropout(rate=0.5))
model.add(Dense(1, activation='sigmoid'))


model.compile(loss='categorical_crossentropy',
              optimizer =Adam(lr=LEARNING_RATE), metrics = ['accuracy'])

model.fit(x_train,
          y_train,
          batch_size = BATCH_SZ,
          epochs=EPOCHS,
          validation_data=(x_val, y_val),
          class_weight={0: 1.,
                        1: 2.,
                        2: 2.,
                        3: 1.,
                        4: 1.,})

print("Accuracy: ", model.evaluate(x_test, y_test)[1])
y_pred = model.predict(x_test)
for i in range(10):
    print(y_pred[i], ", ", np.argmax(y_test[i]))

print("\nConfusion Matrix:")
matrix = np.zeros((4, 4))

for i in range(len(y_pred)):
    label = np.argmax(y_test[i])
    pred = np.argmax(y_pred[i])
    matrix[label][pred]+=1

print("aasavari:\t", matrix[0])
print("bhairav:\t", matrix[1])
print("bilawal:\t", matrix[2])
print("khamaaj:\t", matrix[3])
print("yaman:\t", matrix[4])
```